科学计算与编程基础项目报告

**姓名：王若溪**

**学号：10205000460**

**日期**：**2021/12/31**

**项目名称：基于CO2和GHG相关指标数据分析全球碳排放情况**

[科学计算与编程基础项目报告 1](#_Toc30897)

[项目目标和任务 2](#_Toc28384)

[项目实现的方法和步骤 3](#_Toc15923)

[项目结果呈现 3](#_Toc19015)

[可视化部分： 3](#_Toc18637)

[1.将图表的分析逻辑绘制成树形图： 3](#_Toc29222)

[2.绘制出8张静态图表，并将其和树状图以网页形式汇总到一个界面中： 4](#_Toc8138)

[3.绘制出人均GHG排放量动态图（随时间）： 10](#_Toc4230)

[4.绘制出中国不同部门GHG排放量3D图（随时间）： 10](#_Toc11599)

[5.绘制出全球GHG排放3D地图： 11](#_Toc13868)

[建模与数据分析： 11](#_Toc14683)

[1.数据整体特征查看 11](#_Toc25639)

[2. 13个因素分别和碳排放量做散点图，看大致相关性以及趋势（仅展示部分） 12](#_Toc23976)

[3. 筛选主要影响因素 12](#_Toc32424)

[4.构建不同模型，比较拟合结果 16](#_Toc16789)

[5.预测气温变化规律： 19](#_Toc23476)

[项目的总结 22](#_Toc9873)

[创新点： 22](#_Toc29823)

[不足之处： 22](#_Toc20629)

[改进方向： 22](#_Toc13925)

[代码 24](#_Toc6365)

[可视化部分： 24](#_Toc22186)

[描述性分析： 38](#_Toc2012)

[因子分析： 41](#_Toc7273)

[建模： 44](#_Toc28292)

[北半球气温预测： 52](#_Toc28763)

**项目背景**

上世纪中叶以来，随着世界工业的蓬勃发展，人类活动的频繁密切，大量矿物燃料持续燃烧以及建筑工业的迅猛兴起，排放到大气中的二氧化碳也与日剧增。据科学估算，大气中的二氧化碳含量每增加一倍，全球均温将上升2.9℃。而仅仅从在上世纪初到70年代末，二氧化碳排放量就增加了将近160亿吨，相当于整个大气中二氧化碳总量的0.6%，更不用说近年来密集的工厂兴建和交通所致的大量燃油消耗，进一步加剧了二氧化碳“过排放”。由此带来的温室效应已经在给人类的生产生活带来着不可估量和难以解决的影响。海平面升高，酸雨频繁发生，雨林面积锐减，这些都是时时刻刻发生在我们身边的，发生在人类目前唯一家园上的不可忽视的创伤。

基于此，我们决定对二氧化碳的各个角度进行一个尽可能全面的分析。包括排放来源，各国二氧化碳人均排放量，全球年均二氧化碳增量，预测碳排放量变化趋势以及气温变化趋势等，希冀通过对过往几十年来各地区的二氧化碳数值的多角度分析，能够充分全面的认识到目前碳排问题如此严重的背后原因，也希望通过对趋势的预测，能够在警醒的同时提出一些可能的缓解措施。

**项目目标和任务**

任务：

1.输入基于GHG和CO2的各方面数据，将其分别转化为可视化图表，以更直观清晰的方式显现数据背后蕴藏的与碳排放直接或间接相关的信息，充分了解与二氧化碳排放有关的各项指标，它们对碳排放的“贡献”大小，全球不同国家不同年份的碳排量等。

2.以中国为例，根据数据，分析其与碳排放之间的相关性，找出最直接影响了碳排放的几大因素，并对这些因素的实际社会情况进行重点关注。

3.基于已知数据以及分析得出的重要影响因素，构建不同的模型进行拟合预测，对比不同模型的拟合效果，并提出一些缓解措施。

4.通过时序网络预测气温变化规律，并与实际气温变化进行对比。

项目实现的方法和步骤

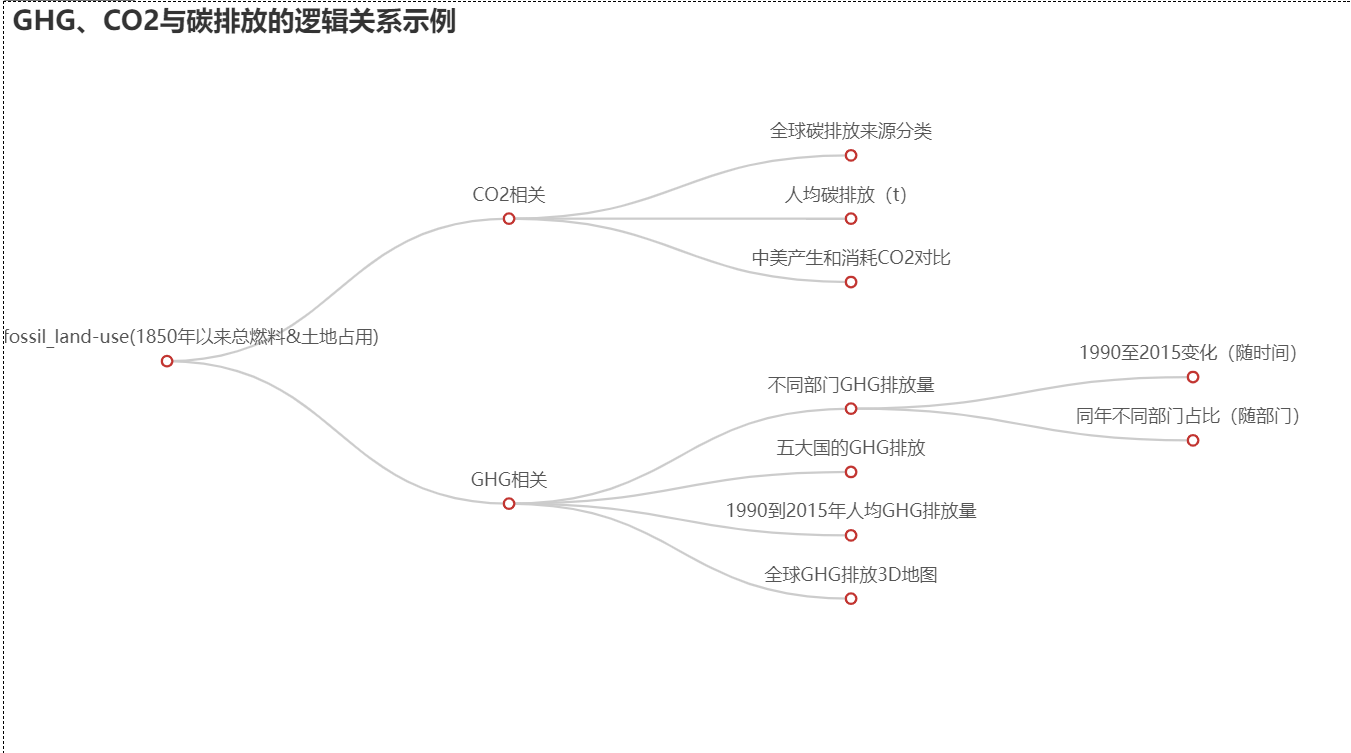
使用算法：python

1. 收集并处理数据。在统计年鉴，CEIC数据库等官方权威网站上搜集数据，下载并整理为Excel表格。使用模块：pandas,sklearn,numpy,keras,datetime,matplotlib,statsmodels
2. 图表绘制。利用收集到的数据编写程序绘制图表并进行图表分析。使用模块：pyecharts
3. 相关性分析。首先确定13个可能的影响因素，主要有GDP，能源消耗量，汽车数量等。然后通过excel制成数据矩阵，并将数据矩阵导入jupyter。再进行描述性分析。查看数据的整体特征，通过matplotlib绘制某因素和碳排放之间的散点图观察大致的变化趋势，计算相关系数矩阵，通过KMO检验和Bartlett球形检验，决定使用因子分析，用相关系数矩阵的特征值筛选主要影响因素。
4. 模型预测。构建多元线性回归，高次线性回归，人工神经网络等不同模型，并考察不同模型的拟合效果。
5. 气温变化预测。运用statsmodels模块中的ExponentialSmoothing方法，拟合模型，并用可视化方式输出预测的气温变化规律。

项目结果呈现

**可视化部分：**

1.将图表的分析逻辑绘制成树形图：

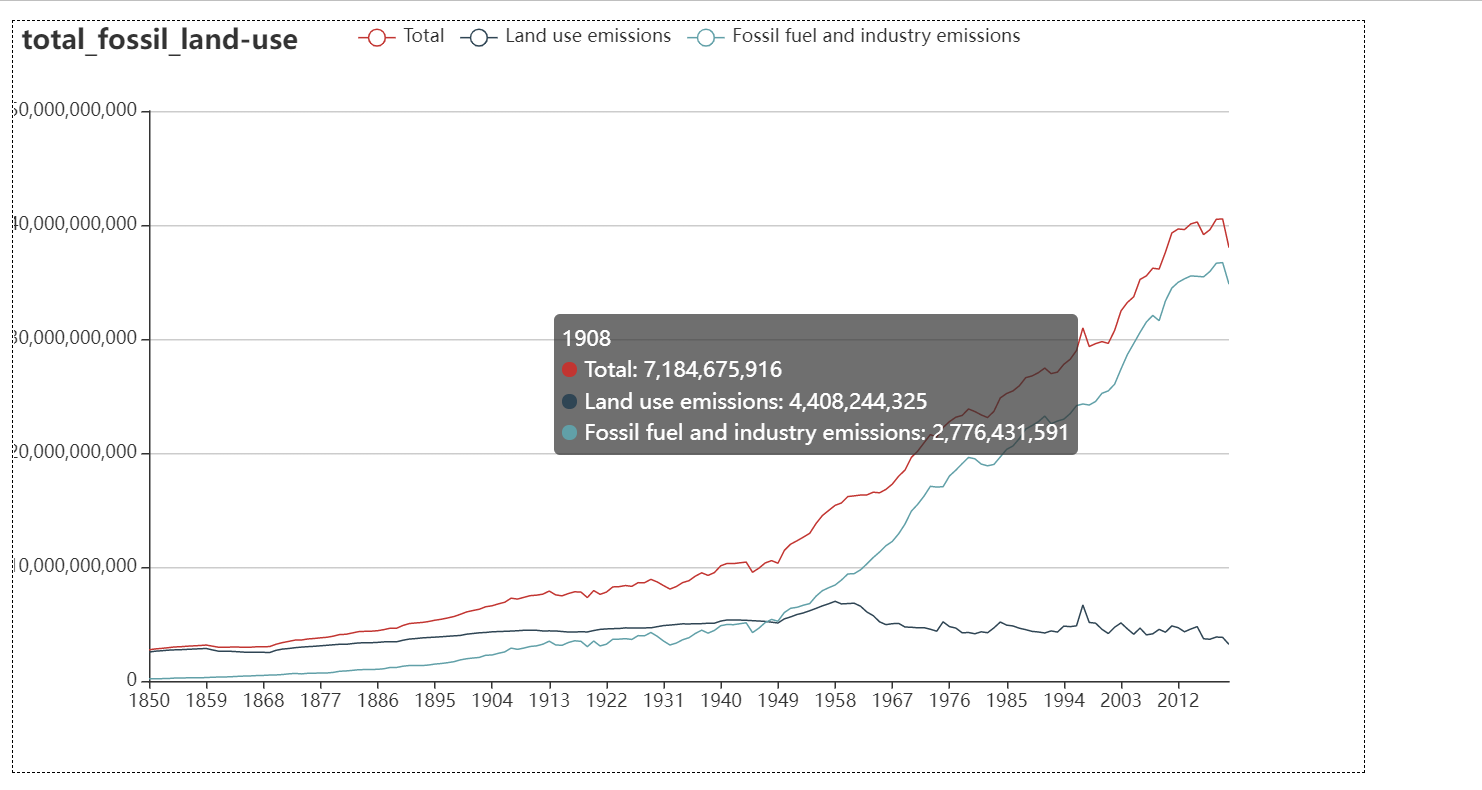


基于1850年以来燃料消耗和土地占用而产生大量温室气体（CO2为主）的情况，我们分两条线分析碳排放相关情况及GHG相关情况。

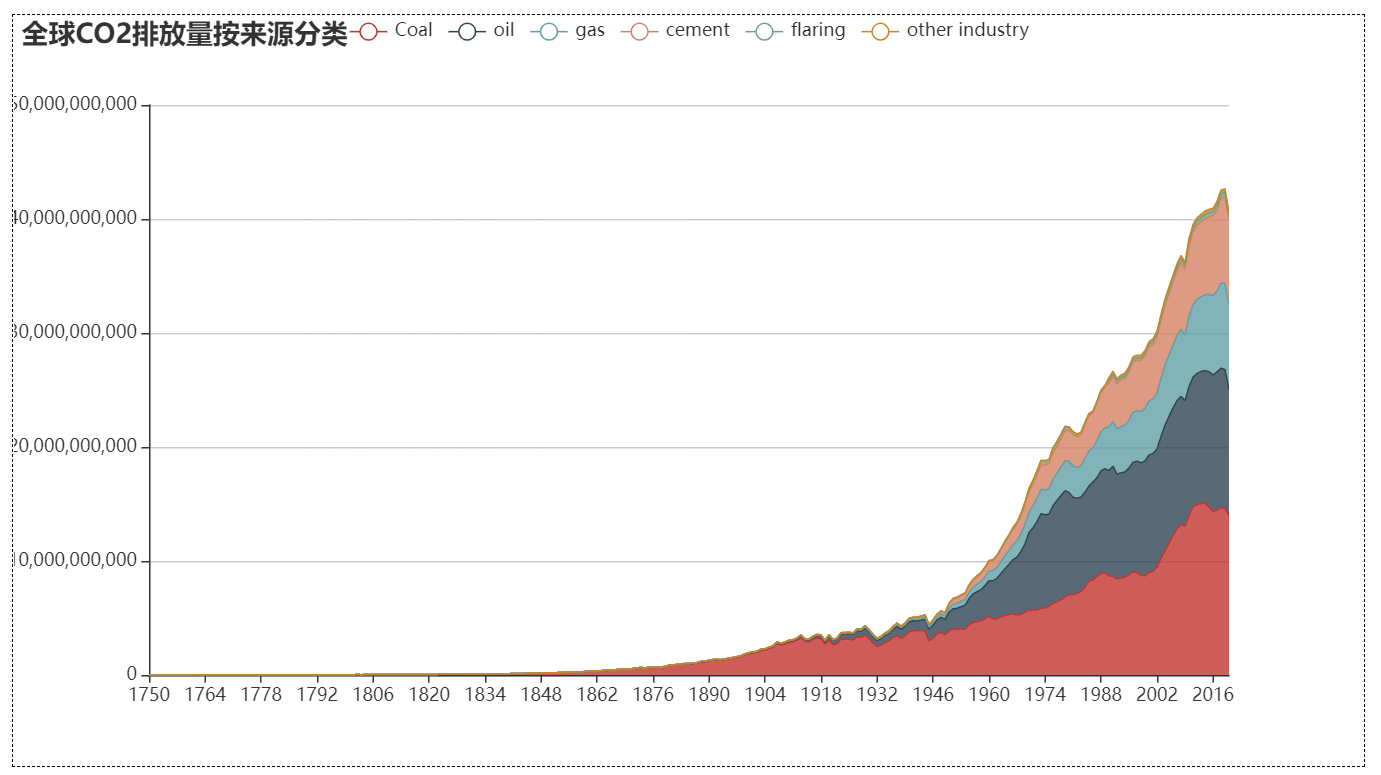
碳排放从全球排放来源，人均排放量（吨），中美两国消耗和产生量对比关系三个角度比较分析。

GHG排放从不同部门排放量（随时间及随部门），五大国GHG排放，1990-2015年人均GHG排放量，全球GHG排放3D地图角度出发分析。

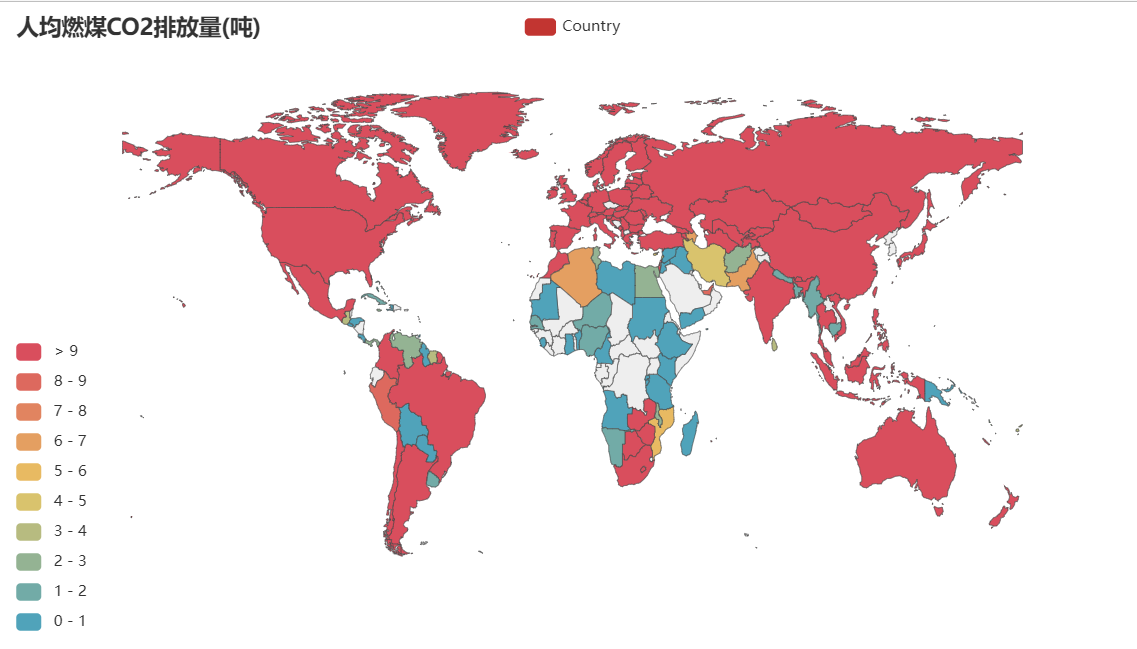
2.绘制出8张静态图表，并将其和树状图以网页形式汇总到一个界面中：



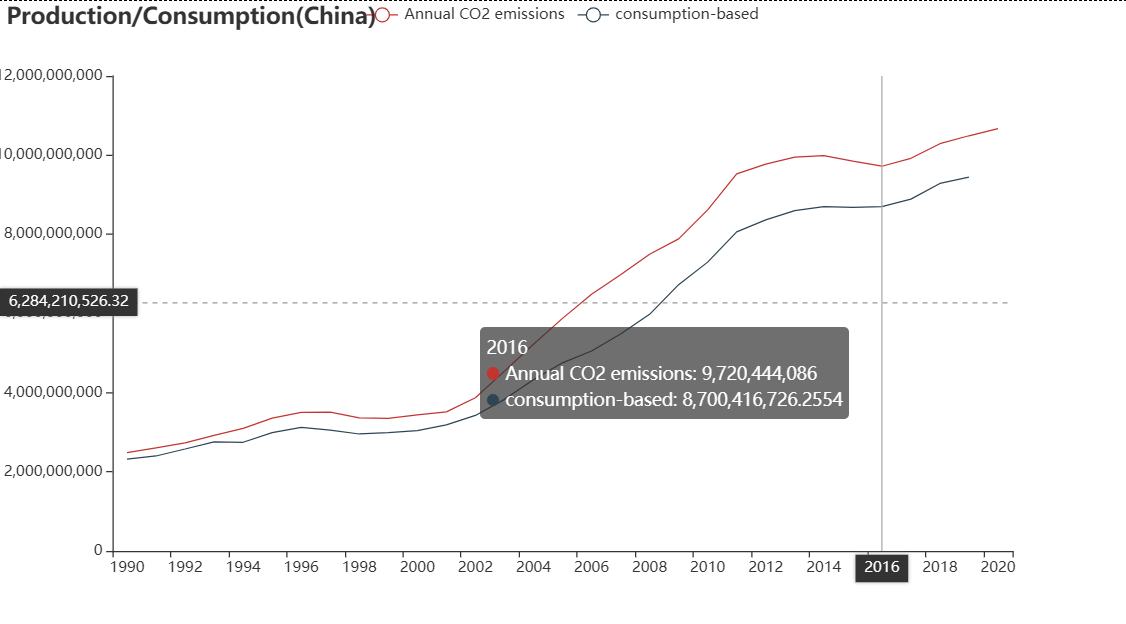
该图时间区间设定在1850年以来，重点分析随年份变化土地利用导致的碳排放，燃料消耗导致的碳排放，总碳排放的数值变化。可以看出土地利用产生的二氧化碳量一直维持在相对稳定的状态，但燃料消耗在1949年后有了一个较大幅度的提升，这也和20世纪以来全球工业水平迅猛发展有直接关系。同时，总排放量也随着燃料排放的增加其有了相同趋势的增幅。

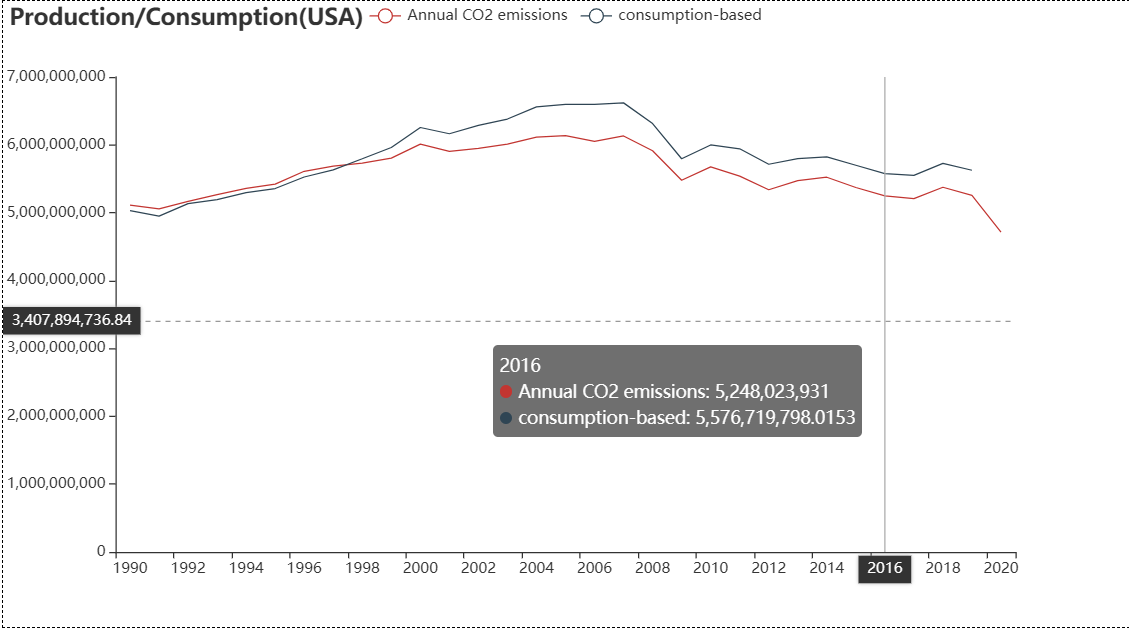


该图时间线设定在1750-2016年。分析了全球碳排放来源的分类。可以看到以煤炭石油天然气等为主的化石燃料对碳排放起到了主要作用。其中煤炭燃料的排放占比稳居最大值，排放量在60年代以前都相对处于低水平，但60年代往后各燃料的碳排放量都有了较大幅度的上升，近十年表现更为明显。



该图为人均燃煤碳排放量（吨）以全球地图平铺图为基础，以国家为指标划分，计算统计了各国人均碳排放量的大小关系。颜色越偏红色的碳排放量越大。可以看出美国加拿大，澳大利亚，中国，俄罗斯等国家的碳排放位于全球前列。



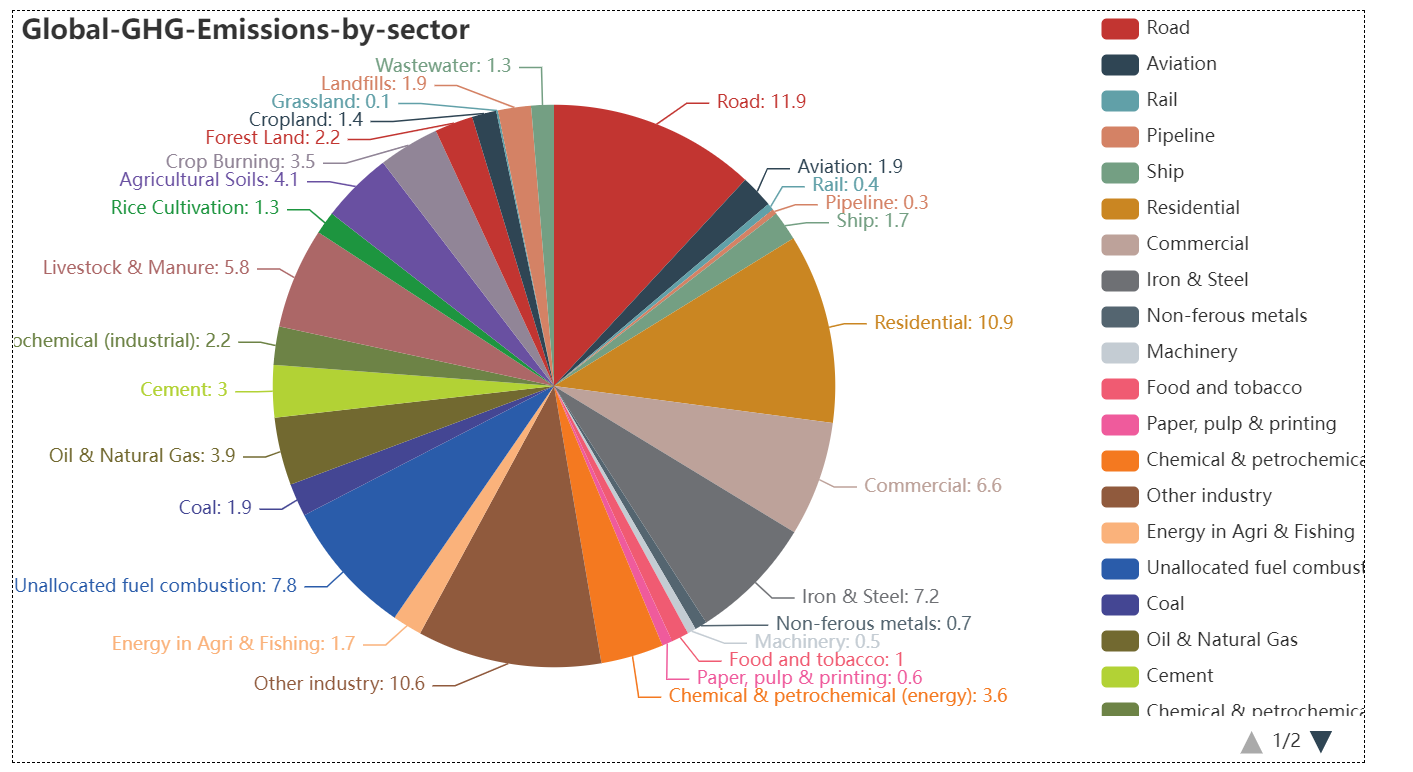


当各国比较二氧化碳排放量时，他们往往关注基于生产的排放--在一个国家自己的边界内所排放的二氧化碳。然而，这未能捕捉到贸易商品的排放--在为其他地方生产商品时排放的二氧化碳。我们可以通过校正贸易来估计基于消费的二氧化碳排放量。

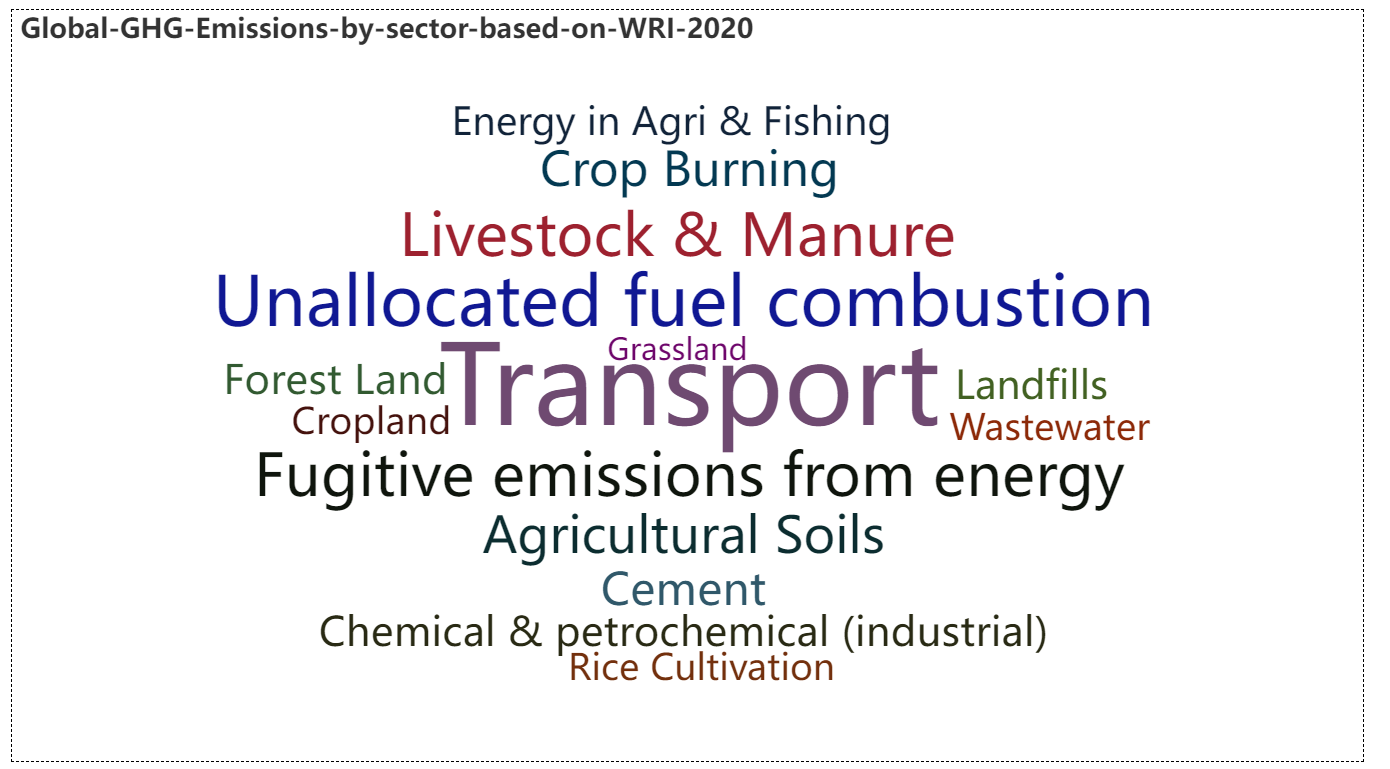
从第二张图可以看到，美国自1998年后，基于消费的排放高于基于生产的排放。2016年，这两个数值分别为56亿和53亿吨--相差8%。这说明美国进口的商品在生产过程中排放的二氧化碳比美国人出口的产品多。中国的情况正好相反：2016年，基于消费的排放量比基于生产的排放量低14%。

这说明虽然中国是一个大型的二氧化碳排放出口国，但它不再是一个大型的排放国，因为它为世界其他国家生产商品。

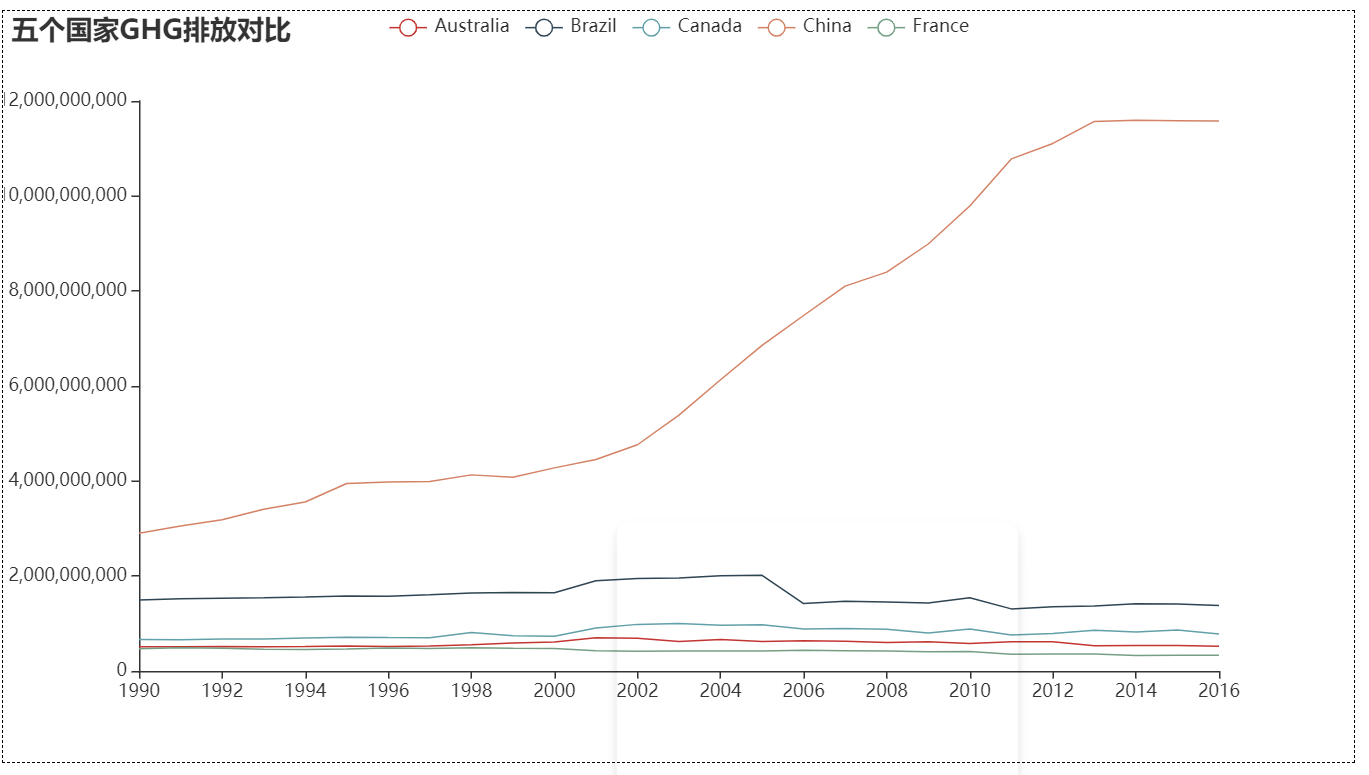
这些比较提供了一个问题的答案，即各国是否只是通过将排放密集型生产离岸到其他国家来实现减排。如果只有基于生产的排放在下降，而基于消费的排放在上升，这就表明是把排放'离岸'到了其他地方，该国家实际上并没有为全球减排做出努力，例如美国。



该图呈现了全球范围内不同产业部门对GHG排放量的贡献占比情况。可以看出道路，住宅，商业，铁矿，牲畜畜牧等行业对GHG排放的贡献率很大。

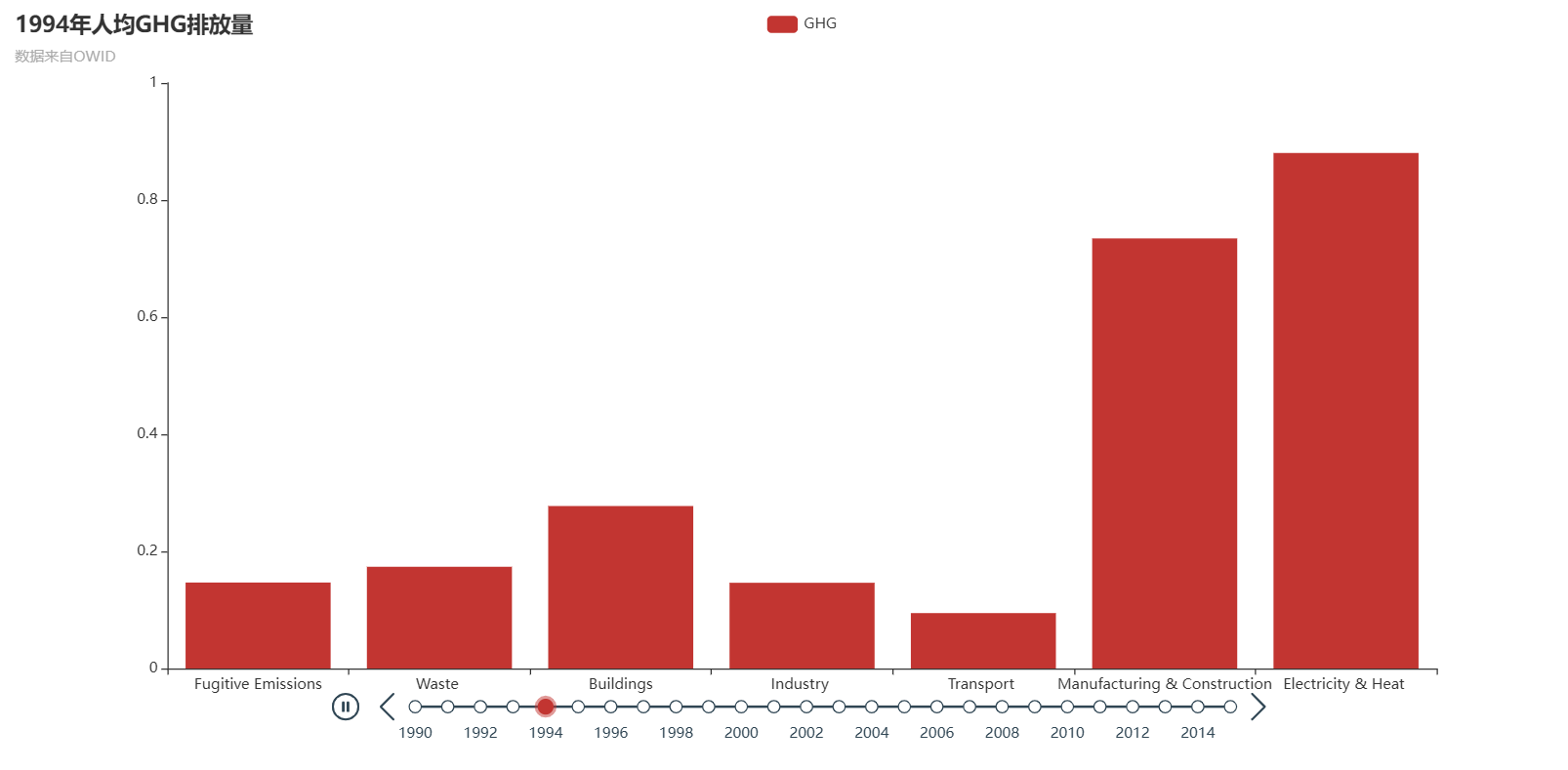


该图以词云图的形式展现了不同部门对于GHG排放的贡献大小。单词越大的贡献率越高。可以看到交通部门稳居首位。其次是牲畜粪肥和燃料燃烧等。



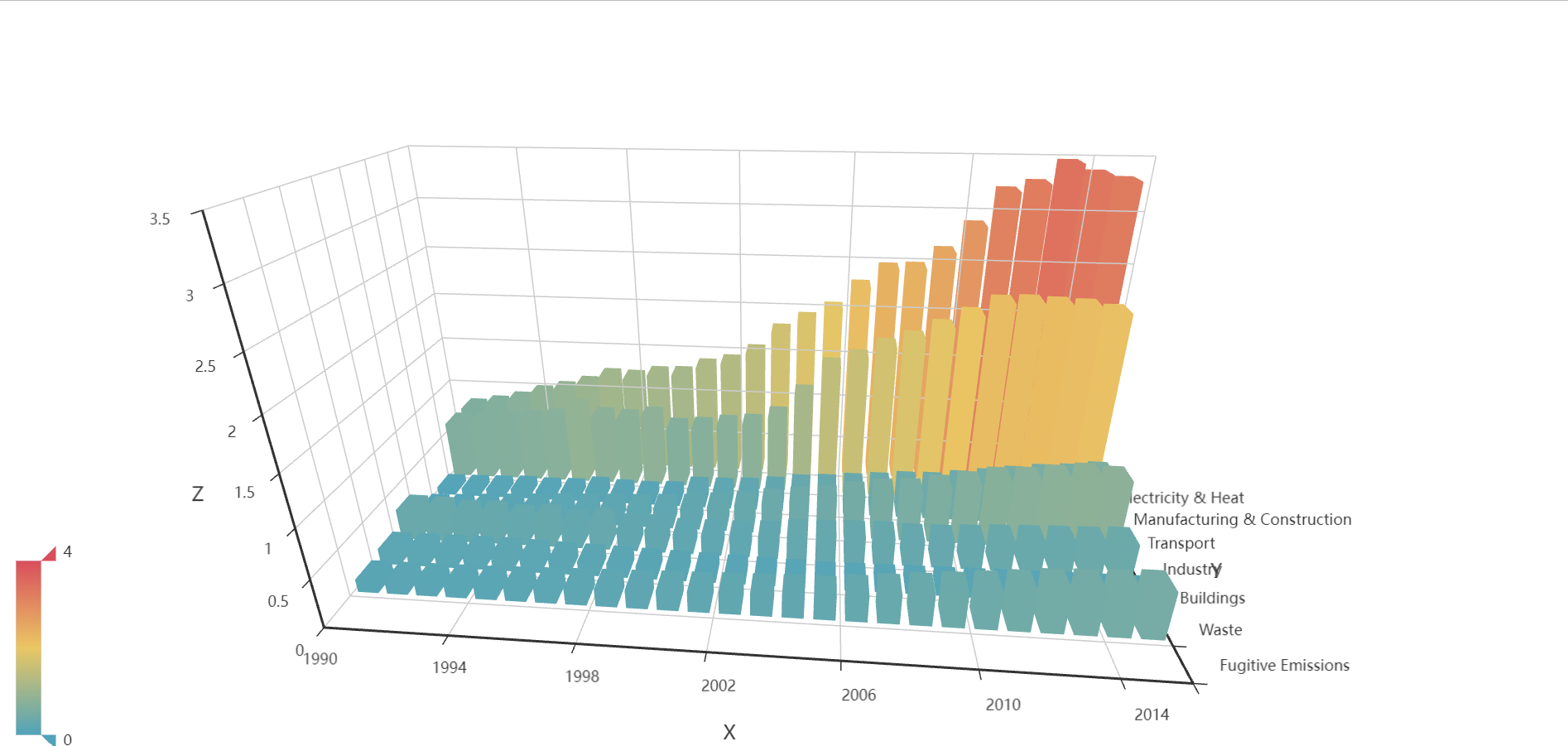
该图比较了世界五个主要的代表性国家对GHG从1990-2016年的排放量。可以看到澳大利亚的GHG排放一直超过其他四国，并在2000年以来有了较大比重的增幅。反观其他四国，大都维持在一个相对稳定趋同的水平。

3.绘制出人均GHG排放量动态图（随时间）：



该图给出了从1990-2015年的人均GHG排放量动态变化。可以看出电力能源和建筑交通方面对GHG排放有很大影响。

4.绘制出中国不同部门GHG排放量3D图（随时间）：



该图以3D结构形式展现中国不同部门从1990-2015年的GHG排放量从柱形图高度上可以得出与上一张图相似的结论，但大小关系比较比上一张图更加直观明显。

5.绘制出全球GHG排放3D地图：



该图以3D地球仪的形式展现了全球各个国家的GHG排放量。颜色越深越接近暖色调的排放量越大。

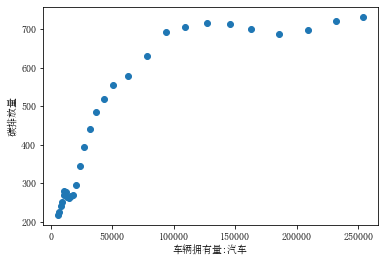
**建模与数据分析：**

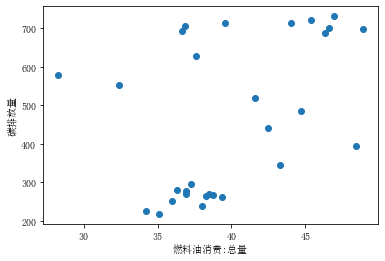
1.数据整体特征查看



本表格对加上碳排放量数据的14个因素指标进行了描述性分析。具体分析了总数，均值，标准差，最值，分位数。

1. 13个因素分别和碳排放量做散点图，看大致相关性以及趋势（仅展示部分）

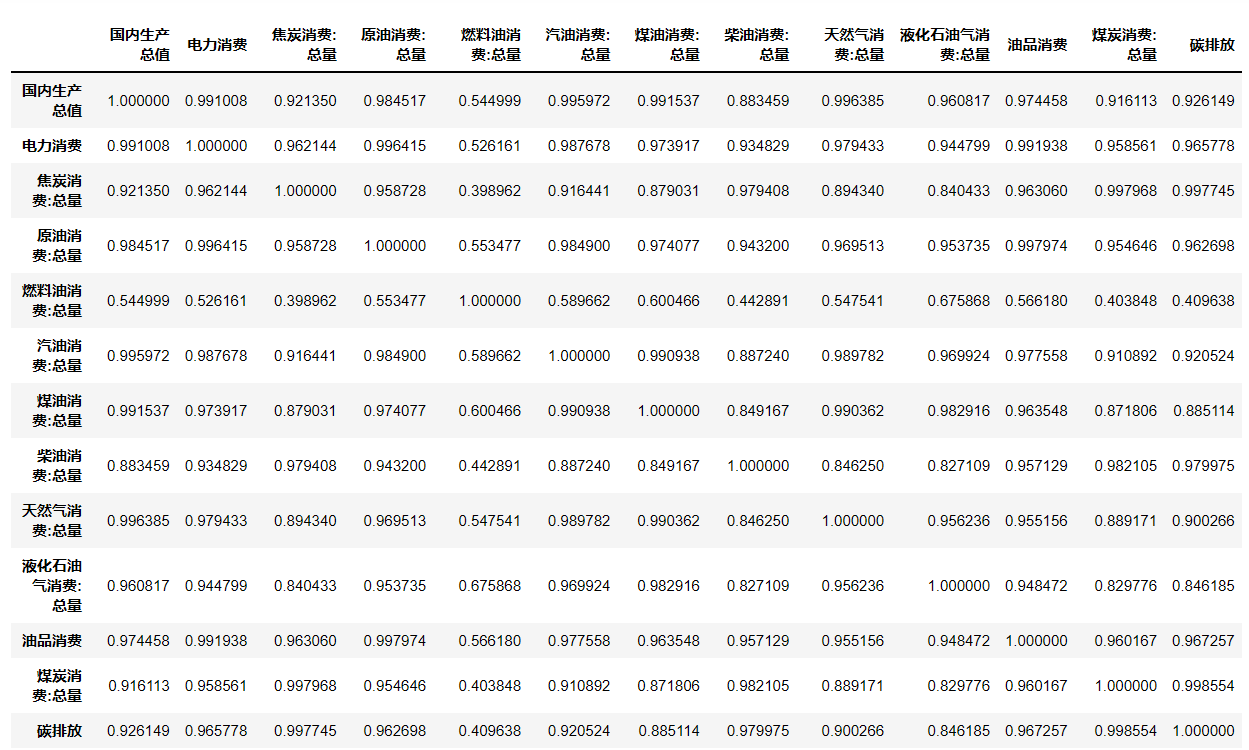




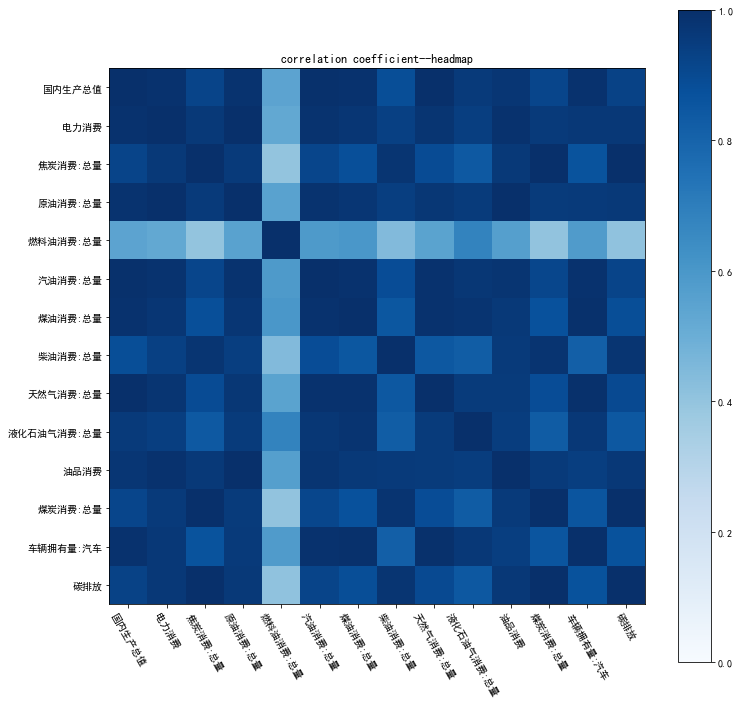
此处展现了碳排放分别与汽车拥有量和燃料消费总量的相关性比较图。可以看出汽车拥有量的逐年增加牵动了碳排放的增加，二者之间相关性较强。反观燃料消费总量，点与点之间散装分布没有规律，其与碳排放之间的相关性不强。

1. 筛选主要影响因素

相关系数矩阵：



相关系数矩阵热力图：



KMO检验：KMO测度为0.844701210761193，大于0.8表明适合因子分析，即变量之间的相关性强，可以从变量群中提取共性因子，对数据降维处理。

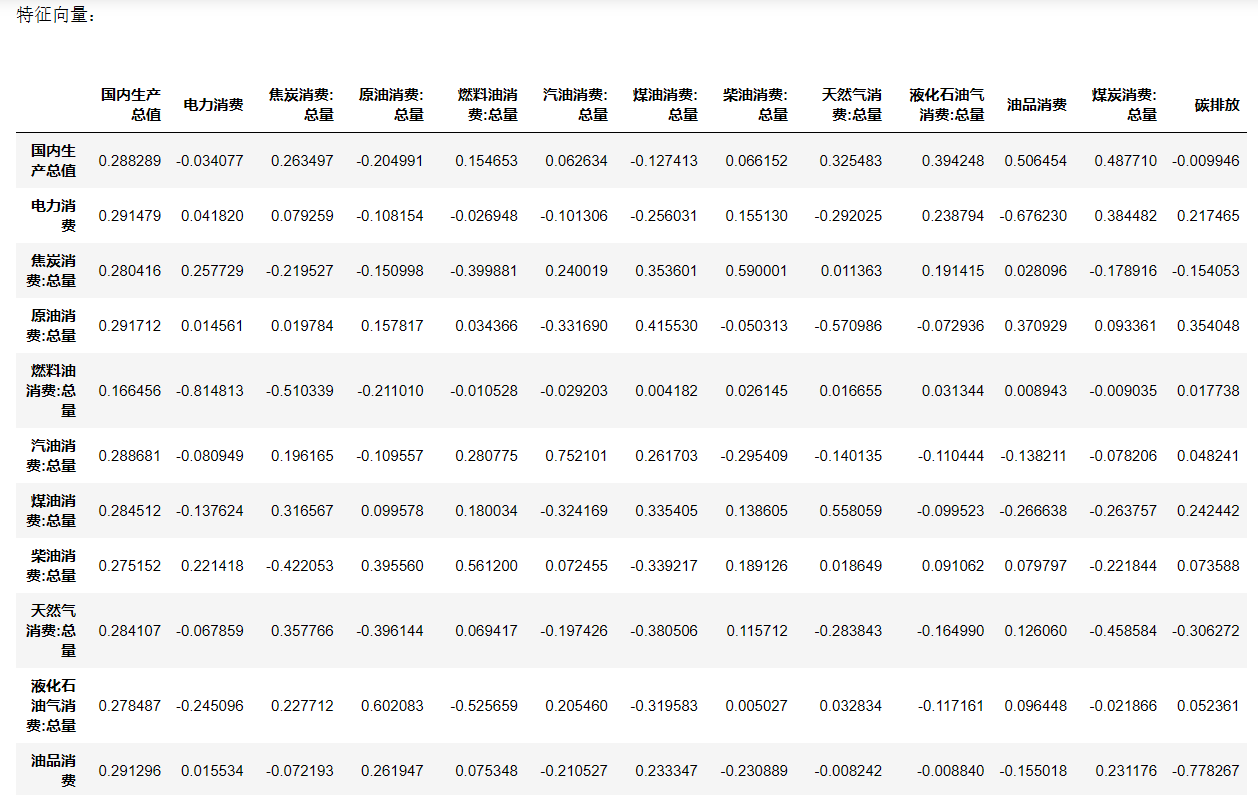
Bartlett球形检验：p值大于0.05，拒绝原假设，所以数据之间不是彼此独立的，可以使用主成分分析。



根据上述两个检验，我们可以对选中的13个变量进行因子分析，先求出相关系数矩阵的特征值和特征向量。

特征值与特征向量：





使用前m个特征值的比重大于99%的标准，我们选出了三个公共因子，国内生产总值、电力消费和焦炭消费。





求出选中的三个因子的因子载荷阵和总方差，第一个因子解释了总体方差的90.1185%，三个公共因子的方差贡献率为99.4469%，可以较好地解释总体方差。

4.构建不同模型，比较拟合结果

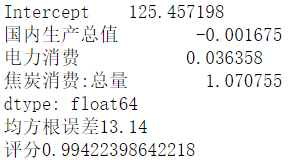
首先进行基础的线性回归拟合模型，并给效果打分。

多元线性回归（未筛选公因子）



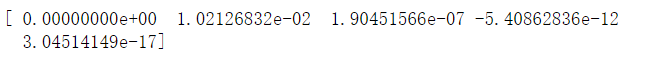
首先拆分数据集，二八分配测试集和训练集，再采用sklearn中的LinearRegression方法构建13个自变量和碳排放量的模型。拟合结果如上图所示。系数为正为正相关，系数为负则为负相关。但拟合出的相关性和描述性分析时得出的直观结果有所不同，直观结果显示除了燃料油消费之外，其他因素基本和碳排放呈正相关，但拟合结果中呈现负相关的量很多。究其原因可能是自变量因素之间存在相关性，自变量与碳排放之间关系有强弱，且非简单的一次线性关系。采取多元一次线性回归虽然最后评分较高，但是拟合结果仍然和实际存在较大偏差。

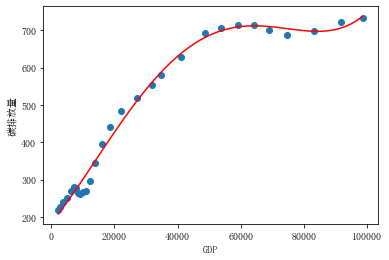
（筛选公因子之后）



筛选公因子之后再做一次多元一次线性回归，仍然会出现负相关情况。一方面GDP和碳排放之间可能并非简单的一次线性关系，另一方面近些年来国家采取有效措施，确使碳排放量的增长有所改观。

高次线性回归（GDP-碳排放）

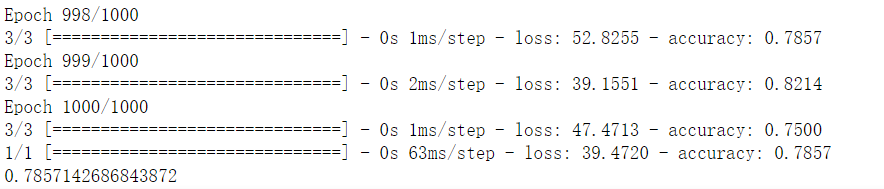




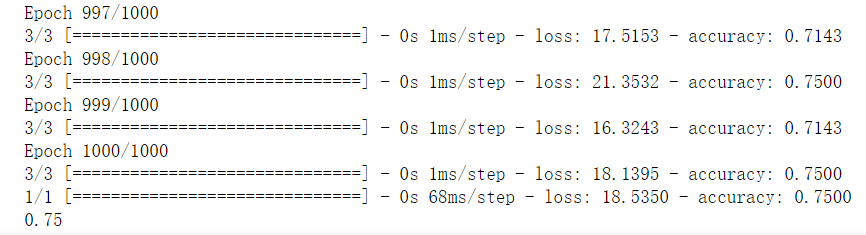
单独将GDP和碳排放量进行高次拟合，拟合结果表明GDP和碳排放之前确实存在高次线性关系。

线性回归之外，我们还采取了其他几种机器学习方法，将多种方法的效果进行比较：

Sequential模型效果打分（未筛选公因子）

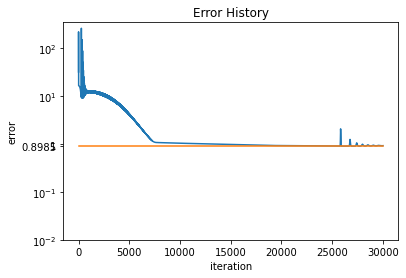


（筛选公因子）



为了提高模型准确率，采用0-1分类，处理碳排放数据，逐年做差，相较于上一年增长的订标签为1，下降的订标签为0。导入keras库中的Sequential方法构建模型，分别训练13个因子和3个因子1000次，准确率均大于0.75，效果较好。

BP神经网络模型误差：



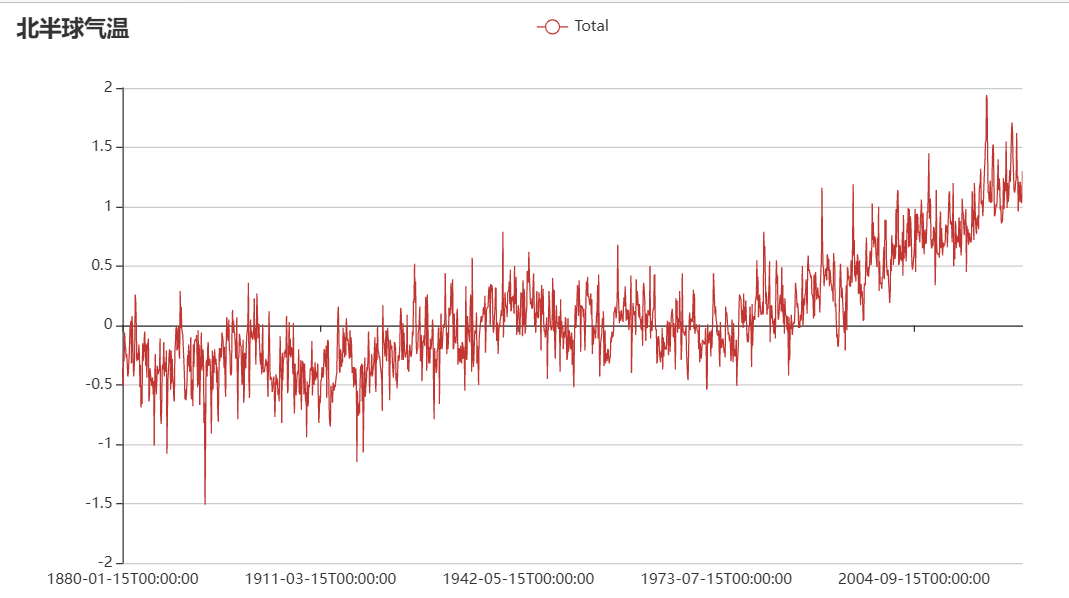
参考网上手撕代码构建神经网络。不进行0-1分类，3个主因子为输入层，碳排放量为输出层，设置8个隐藏层神经元，训练30000次，最终用可视化方式给出训练过程中的误差曲线。误差降到1一下，效果也还不错。

对比采用的三种模型，拟合预测效果均不错。

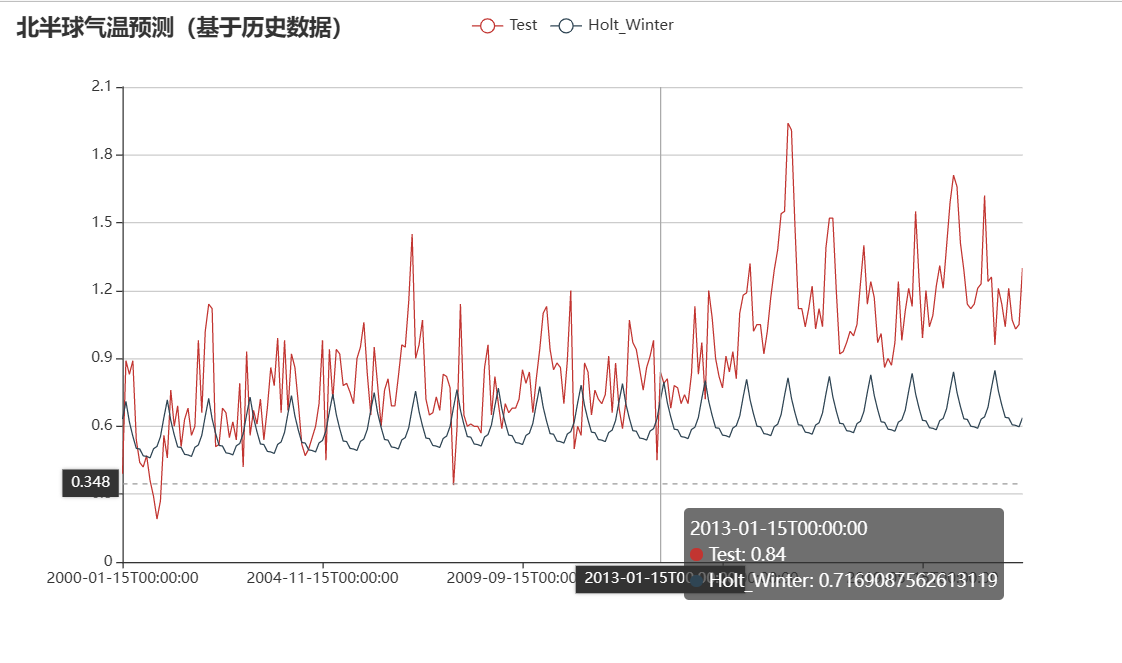
5.预测气温变化规律：

考虑时间序列分析中的指数平滑算法，指数平滑法不舍弃过去的数据，但是仅给予逐渐减弱的影响程度，即随着数据的远离，赋予逐渐收敛为零的权数。我们尝试使用该方法对未来北半球的气温变化进行预测。

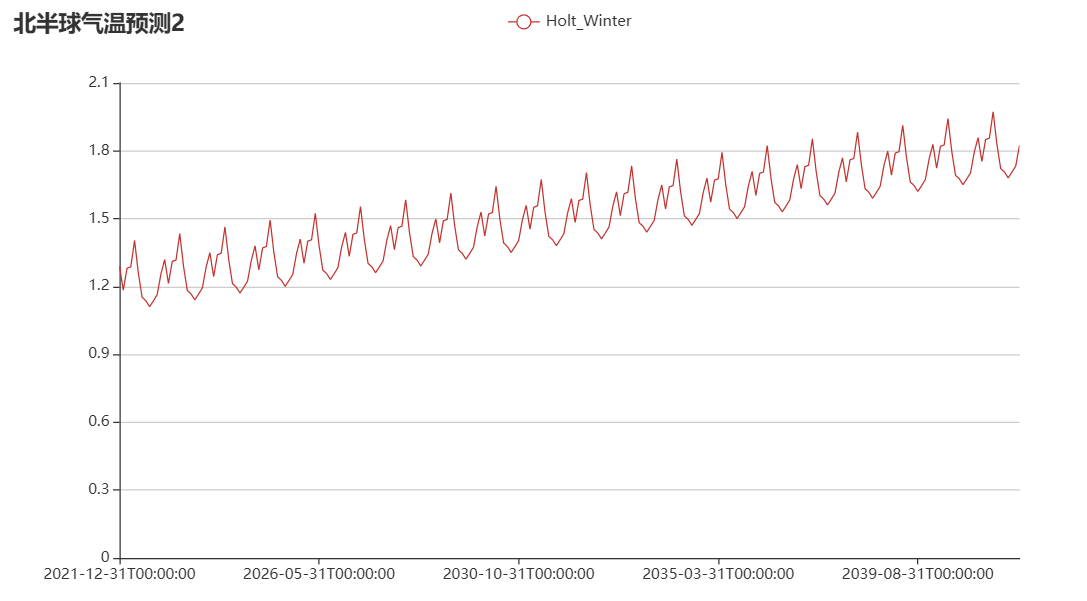
用1880年至2000年每月的平均气温变化量作为训练集，运用statsmodels模块中的ExponentialSmoothing方法，拟合模型，并用可视化方式输出预测的气温变化规律。（图为1880年至2020年北半球气温逐月变化量）。



用训练好的模型尝试预测2000年至2021年的气温变化，由于近年来全球变暖形势严峻，可以看到模型对2000年至2013年的气温变化预测基本符合，但2013年后北半球的逐月气温变化量在0.9℃到2.0℃之间，训练集中的数据在0℃附近波动，差异较大，因为该模型没有将人为因素作为变量（比如近年来工业发展导致的碳排放量大幅增加），所以预测结果并不理想。



考虑到上述情况，我们尝试使用2000年至2021年9月的气温数据作为训练集，预测2021年12月至2041年12月的逐月气温变化趋势。可以看出如果我们对全球碳排放量的控制，继续按照近20年的生产方式和处理模式对待CO2排放，那么全球变暖将进一步加剧。



项目的总结

（归纳总结项目的工作和创新点以及不足和进一步的改进方向）

创新点：

1.图表互动性。与一张单纯只能看的图表不同，我们的图表当鼠标移动其上的每一个位置时都能随之显示该点的数据情况，对应的年份国家，方便在直观感受变化趋势的时候也能同步关注到数值情况，两者配合分析会更精确更严谨.

2.通过时序模型拟合预测气温变化规律，并与实际气温变化进行比较。

3. 3D图表&动态图表呈现。为了便于比较全世界各国的GHG排放量，我们在做好一张地图平铺图的基础上又做出了一个3D地球仪式地图，可拖拽转动，放大缩小。能更直观有趣的看出排放量在全球的分布情况。其次，为看出随年份变化GHG的动态变化，我们做出一张1990-2015年GHG变化量动态图表，图表各项数值随时间推移自动跳动。

4.折叠层树状图梳理所有表格的逻辑关系。为了将多张表格之间的关系汇总到一起，我们做了一个向右延展的tree map，其中更细的层级部分可以通过隐藏点的点击打开展现。层次清楚条理清晰。

不足之处：

1.在对产生二氧化碳的因素与碳排放之间的相关性分析上，收集到的数据较少。大部分只收集到了近几年的相关数据，为了统一时间利于比较分析，所有数据均只取了近十年的相关部分。与现实情况联系密切但长久分析不够充分。无法反映从上世纪开始一路而来的变化情况，在时间跨度上比较局限。

2.多选用现有模型进行拟合，虽然拟合效果还行，但并未能对现有模型进行进一步改进。

3.对于预测碳排放未来数据方面并未找到合适的模型。

改进方向：

1.制表时可以不用完全统一时间刻度。当数据收集的时间指标上有参差时，不用废弃多余数据，也可考虑将其他变量作为固定量以从另外的角度分析数据之间的关系，例如当石油消耗量收集到了近二十年数据，柴油消耗量收集到了近二十年的数据，二氧化碳排放量收集到了近十年的数据时，除了分析近十年石油柴油对碳排的贡献率大小，可以考虑将柴油石油归并为一类“交通工具燃油”，对比交通工具燃油前后十年消耗量的变化趋势。做到多角度充分利用数据。

代码

**可视化部分：**

from pyecharts import options as opts

from pyecharts.charts import Bar, Grid, Line, Liquid, Page, Pie,Map,Timeline,Bar3D,WordCloud,Tree

from pyecharts.commons.utils import JsCode

from pyecharts.components import Table

from pyecharts.faker import Faker

from pyecharts.globals import ThemeType

from pyecharts.charts import MapGlobe

from pyecharts.faker import POPULATION

from pyecharts.globals import SymbolType

import pandas as pd

def tree():

data = [

{

"children": [

{

"children": [{ "name": "全球碳排放来源分类"}, {"name": "人均碳排放（t）"},{"name": "中美产生和消耗CO2对比"}],

"name": "CO2相关",

},

{

"children": [

{"children": [{"name": "1990至2015变化（随时间）"}, {"name": "同年不同部门占比（随部门）"}], "name": "不同部门GHG排放量"},

{"name": "五大国的GHG排放"},{"name": "1990到2015年人均GHG排放量"},{"name": "全球GHG排放3D地图"},

],

"name": "GHG相关",

},

],

"name": "total\_fossil\_land-use(1850年以来总燃料&土地占用)",

}

]

c = (

Tree()

.add("", data)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="GHG、CO2与碳排放的逻辑关系示例"))

)

return c

def total\_fossil\_land\_use():

data=pd.read\_csv('global-co2-fossil-plus-land-use.csv')

c = (

Line()

.add\_xaxis(data['Year'])

.add\_yaxis("Total",data['Fossil fuel + land use emissions (GtCO2)'] ,symbol\_size= 0)

.add\_yaxis("Land use emissions",data['Land use emissions (GtCO2)'],symbol\_size= 0)

.add\_yaxis("Fossil fuel and industry emissions",data['Fossil fuel and industry emissions (GtCO2)'],symbol\_size= 0)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="total\_fossil\_land-use"),

tooltip\_opts=opts.TooltipOpts(

is\_show = True,

trigger="axis",

axis\_pointer\_type="cross"

),

yaxis\_opts=opts.AxisOpts(

type\_="value",

axistick\_opts=opts.AxisTickOpts(is\_show=True),

splitline\_opts=opts.SplitLineOpts(is\_show=True),

),

xaxis\_opts=opts.AxisOpts(

type\_="category",

boundary\_gap=False,

),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

)

gird=(

Grid()

.add(c,grid\_opts=opts.GridOpts(pos\_left="15%"))

)

return c

def CO2\_by\_source():

df=pd.read\_csv('CO2-by-source.csv')

world\_data=df.loc[df['Entity'] == 'World']

x\_data =world\_data['Year']

c=(

Line()

.add\_xaxis(xaxis\_data=x\_data)

.add\_yaxis(

series\_name="Coal",

stack="总量",

y\_axis=world\_data['Annual CO2 emissions from coal (zero filled)'],

areastyle\_opts=opts.AreaStyleOpts(opacity=0.8),

label\_opts=opts.LabelOpts(is\_show=True,position="right"),

symbol\_size= 0,

)

.add\_yaxis(

series\_name="oil",

stack="总量",

y\_axis=world\_data['Annual CO2 emissions from oil (zero filled)'],

areastyle\_opts=opts.AreaStyleOpts(opacity=0.8),

label\_opts=opts.LabelOpts(is\_show=True,position="right"),

symbol\_size= 0,

)

.add\_yaxis(

series\_name="gas",

stack="总量",

y\_axis=world\_data['Annual CO2 emissions from gas (zero filled)'],

areastyle\_opts=opts.AreaStyleOpts(opacity=0.8),

label\_opts=opts.LabelOpts(is\_show=True,position="right"),

symbol\_size= 0,

)

.add\_yaxis(

series\_name="cement",

stack="总量",

y\_axis=world\_data['Annual CO2 emissions from gas (zero filled)'],

areastyle\_opts=opts.AreaStyleOpts(opacity=0.8),

label\_opts=opts.LabelOpts(is\_show=True,position="right"),

symbol\_size= 0,

)

.add\_yaxis(

series\_name="flaring",

stack="总量",

y\_axis=world\_data['Annual CO2 emissions from flaring (zero filled)'],

areastyle\_opts=opts.AreaStyleOpts(opacity=0.8),

label\_opts=opts.LabelOpts(is\_show=True,position="right"),

symbol\_size= 0,

)

.add\_yaxis(

series\_name="other industry",

stack="总量",

y\_axis=world\_data['Annual CO2 emissions from other industry (zero filled)'],

areastyle\_opts=opts.AreaStyleOpts(opacity=0.8),

label\_opts=opts.LabelOpts(is\_show=True,position="right"),

symbol\_size= 0,

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="全球CO2排放量按来源分类"),

tooltip\_opts=opts.TooltipOpts(trigger="axis", axis\_pointer\_type="cross"),

yaxis\_opts=opts.AxisOpts(

type\_="value",

axistick\_opts=opts.AxisTickOpts(is\_show=True),

splitline\_opts=opts.SplitLineOpts(is\_show=True),

),

xaxis\_opts=opts.AxisOpts(type\_="category", boundary\_gap=False),

)

)

grid=(

Grid()

.add(c,grid\_opts=opts.GridOpts(pos\_left="15%"))

)

return c

def CO2\_emissions\_average():

df=pd.read\_csv('per-capita-co2-by-fuel.csv')

list1 = []

list2 = []

df.loc[df['Year'] == '2020']

x = df['Entity'].tolist()

y = df['Annual CO2 emissions from coal (per capita)'].tolist()

c = (

Map()

.add(series\_name="Country",

data\_pair=[list(z) for z in zip(x, y)],

maptype='world',

name\_map=df['Entity'],

is\_map\_symbol\_show=False)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

.set\_global\_opts(

title\_opts=opts.TitleOpts(title='人均燃煤CO2排放量(吨)'),

visualmap\_opts=opts.VisualMapOpts(

is\_piecewise=True,

type\_ = "color",

split\_number = 10,

pieces=[{"max": 1, "min": 0},

{"max": 2, "min": 1},

{"max": 3, "min": 2},

{"max": 4, "min": 3},

{"max": 5, "min": 4},

{"max": 6, "min": 5},

{"max": 7, "min": 6},

{"max": 8, "min": 7},

{"max": 9, "min": 8},

{"min": 9}]

)

)

)

return c

def CO2\_China():

df=pd.read\_csv('production-vs-consumption-co2-emissions.csv')

data=df.loc[df['Entity'] == 'China']

x = data['Year'].tolist()

x=[str(i) for i in x]

y1 = data['Annual CO2 emissions'].tolist()

y2=data['Annual consumption-based CO2 emissions'].tolist()

c = (

Line()

.add\_xaxis(xaxis\_data=x)

.add\_yaxis("Annual CO2 emissions", y1 ,symbol\_size= 0)

.add\_yaxis("consumption-based", y2, symbol\_size= 0)

.set\_global\_opts(

xaxis\_opts=opts.AxisOpts(

type\_="category",

is\_show = True,

),

tooltip\_opts=opts.TooltipOpts(

is\_show = True,

trigger="axis",

axis\_pointer\_type="cross",

),

title\_opts=opts.TitleOpts(

title="Production/Consumption(China)"

),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

)

gird=(

Grid()

.add(c,grid\_opts=opts.GridOpts(pos\_left="15%"))

)

return c

def CO2\_USA():

df=pd.read\_csv('production-vs-consumption-co2-emissions.csv')

data=df.loc[df['Entity'] == 'United States']

x = data['Year'].tolist()

x=[str(i) for i in x]

y1 = data['Annual CO2 emissions'].tolist()

y2=data['Annual consumption-based CO2 emissions'].tolist()

c = (

Line()

.add\_xaxis(xaxis\_data=x)

.add\_yaxis("Annual CO2 emissions", y1 ,symbol\_size= 0)

.add\_yaxis("consumption-based", y2, symbol\_size= 0)

.set\_global\_opts(

xaxis\_opts=opts.AxisOpts(

type\_="category",

is\_show = True,

),

tooltip\_opts=opts.TooltipOpts(

is\_show = True,

trigger="axis",

axis\_pointer\_type="cross",

),

title\_opts=opts.TitleOpts(

title="Production/Consumption(USA)"

),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

)

gird=(

Grid()

.add(c,grid\_opts=opts.GridOpts(pos\_left="15%"))

)

return c

def Global\_GHG\_by\_sector():

data=pd.read\_excel('Global-GHG-Emissions-by-sector-based-on-WRI-2020.xlsx')

c = (

Pie()

.add(

"",

[

list(z)

for z in zip(

data['Sub-sector'],

data['Share of global greenhouse gas emissions (%)'],

)

],

center=["40%", "50%"],

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="Global-GHG-Emissions-by-sector"),

legend\_opts=opts.LegendOpts(type\_="scroll", pos\_left="80%", orient="vertical"),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))

)

return c

def Global\_GHG\_cloud():

data=pd.read\_excel("Global-GHG-Emissions-by-sector-based-on-WRI-2020.xlsx",sheet\_name="Sub-sector")

words=data.values.tolist()

c = (

WordCloud()

.add("", words, word\_size\_range=[20, 100], shape=SymbolType.DIAMOND)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="Global-GHG-Emissions-by-sector-based-on-WRI-2020"))

)

return c

def country():

data=pd.read\_csv('total-ghg-emissions.csv')

grouped=data.groupby('Entity')

x = grouped.get\_group('Australia')['Year'].tolist()

y1 = grouped.get\_group('Australia')['Total GHG emissions including LUCF (CAIT)'].tolist()

y2 = grouped.get\_group('Brazil')['Total GHG emissions including LUCF (CAIT)'].tolist()

y3 = grouped.get\_group('Canada')['Total GHG emissions including LUCF (CAIT)'].tolist()

y4 = grouped.get\_group('China')['Total GHG emissions including LUCF (CAIT)'].tolist()

y5 = grouped.get\_group('France')['Total GHG emissions including LUCF (CAIT)'].tolist()

c = (

Line()

.add\_xaxis(data['Year'])

.add\_yaxis("Australia",y1 ,symbol\_size= 0)

.add\_yaxis("Brazil",y2,symbol\_size= 0)

.add\_yaxis("Canada",y3,symbol\_size= 0)

.add\_yaxis("China",y4,symbol\_size= 0)

.add\_yaxis("France",y5,symbol\_size= 0)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="5country-ghg-emissions"))

.set\_global\_opts(xaxis\_opts=opts.AxisOpts(is\_scale= True,interval= 30,min\_=1990,max\_=2017))

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

.set\_global\_opts(

tooltip\_opts=opts.TooltipOpts(trigger="axis"),

toolbox\_opts=opts.ToolboxOpts(is\_show=True),

xaxis\_opts=opts.AxisOpts(type\_="category", boundary\_gap=False),

)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="五个国家GHG排放对比")

))

return c

def page\_draggable\_layout():

page = Page(layout=Page.DraggablePageLayout)

page.add(

tree(),

total\_fossil\_land\_use(),

CO2\_by\_source(),

CO2\_emissions\_average(),

CO2\_China(),

CO2\_USA(),

Global\_GHG\_by\_sector(),

Global\_GHG\_cloud(),

country(),

)

page.render("page\_draggable\_layout.html")

if \_\_name\_\_ == "\_\_main\_\_":

page\_draggable\_layout()

data=pd.read\_excel('China-ghg-sector.xlsx')

year = ['1990','1991','1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014','2015']

sector = list(data.columns[1:])

L=[]

for i in range(data.shape[0]):

for j in range(1,data.shape[1]):

l=[i,j,data.iloc[i,j]]

L.append(l)

data = [[d[0], d[1], d[2]] for d in L]

(

Bar3D(init\_opts=opts.InitOpts(width="1200px", height="600px"))

.add(

series\_name="",

data=data,

xaxis3d\_opts=opts.Axis3DOpts(type\_="category", data=year),

yaxis3d\_opts=opts.Axis3DOpts(type\_="category", data=sector),

zaxis3d\_opts=opts.Axis3DOpts(type\_="value"),

)

.set\_global\_opts(xaxis\_opts=opts.AxisOpts(is\_scale= True,interval= 1))

.set\_global\_opts(yaxis\_opts=opts.AxisOpts(is\_scale= True,interval= 1))

.set\_global\_opts(title\_opts=opts.TitleOpts(title="China-ghg-sector"))

.set\_global\_opts(visualmap\_opts=opts.VisualMapOpts(max\_=4)

)

.render("China-ghg-sector\_bar3d\_punch\_card.html")

)

data=pd.read\_excel('China-ghg-sector.xlsx')

name\_list = list(data.columns[1:])

for i in range(len(name\_list)):

name\_list[i]=name\_list[i][:-13]

Year=list(data['Year'])

data\_ghg={}

for i in range(len(Year)):

data\_ghg[Year[i]]=list(data.iloc[i])[1:]

#####################################################################################

def get\_year\_overlap\_chart(year: int) -> Bar:

bar = (

Bar()

.add\_xaxis(xaxis\_data=name\_list)

.add\_yaxis(

"GHG",

data\_ghg[year],

is\_selected=True,

label\_opts=opts.LabelOpts(is\_show=False),

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(

title="{}年人均GHG排放量".format(year), subtitle="数据来自OWID"

),

tooltip\_opts=opts.TooltipOpts(

is\_show=True, trigger="axis", axis\_pointer\_type="shadow"

),

xaxis\_opts=opts.AxisOpts(axislabel\_opts={"interval":"0"},type\_='category')

)

)

return bar

# 生成时间轴的图

timeline = Timeline(init\_opts=opts.InitOpts(width="1300px", height="600px"))

for y in range(1990, 2016):

timeline.add(get\_year\_overlap\_chart(year=y), time\_point=str(y))

timeline.add\_schema(is\_auto\_play=True, play\_interval=1000)

timeline.render("中国人均GHG排放量.html")

data=pd.read\_csv('total-ghg-emissions.csv')

grouped=data.groupby('Year')

x = grouped.get\_group(2016)['Entity'].tolist()

y = grouped.get\_group(2016)['Total GHG emissions including LUCF (CAIT)'].tolist()

low, high = min(y), max(y)

c = (

MapGlobe()

.add\_schema()

.add(

maptype="world",

series\_name="World Population",

data\_pair=[list(z) for z in zip(x, y)],

is\_map\_symbol\_show=False,

label\_opts=opts.LabelOpts(is\_show=False),

)

.set\_global\_opts(

visualmap\_opts=opts.VisualMapOpts(

min\_=low,

max\_=high,

range\_text=["max", "min"],

is\_calculable=True,

range\_color=[ "lightskyblue","yellow", "orangered"],

)

)

.render("total-ghg-emissions\_map\_globe\_3D.html")

)

**描述性分析：**

import pandas as pd

from pandas import DataFrame

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['Simsun']

plt.rcParams['axes.unicode\_minus'] = False

target\_url = ("final(1).xlsx")

#读取数据

data = pd.read\_excel(target\_url,index\_col='年份')

y=data['碳排放']

#描述性分析

print('描述性分析')

display(data.describe())

#绘制交会图-GDP和碳排放

x1=data['国内生产总值']

plot.scatter(x1,y)

plot.xlabel("GDP")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和车辆拥有量:汽车

x2= data['车辆拥有量:汽车']

plot.scatter(x2,y)

plot.xlabel("车辆拥有量:汽车")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和电力消费

x3= data['电力消费']

plot.scatter(x3, y)

plot.xlabel("电力消费")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和焦炭消费:总量

x4= data['焦炭消费:总量']

plot.scatter(x4, y)

plot.xlabel("焦炭消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和原油消费:总量

x5= data['原油消费:总量']

plot.scatter(x5, y)

plot.xlabel("原油消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和燃料油消费:总量

x6= data['燃料油消费:总量']

plot.scatter(x6, y)

plot.xlabel("燃料油消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和汽油消费:总量

x7= data['汽油消费:总量']

plot.scatter(x7, y)

plot.xlabel("汽油消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和煤油消费:总量

x8= data['煤油消费:总量']

plot.scatter(x8, y)

plot.xlabel("煤油消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和柴油消费:总量

x9= data['柴油消费:总量']

plot.scatter(x9, y)

plot.xlabel("柴油消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和天然气消费:总量

x10= data['天然气消费:总量']

plot.scatter(x10, y)

plot.xlabel("天然气消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和液化石油气消费:总量

x11= data['液化石油气消费:总量']

plot.scatter(x11, y)

plot.xlabel("液化石油气消费:总量")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和油品消费

x12= data['油品消费']

plot.scatter(x12, y)

plot.xlabel("油品消费")

plot.ylabel("碳排放量")

plot.show()

#绘制交会图-GDP和煤炭消费:总量

x13= data['煤炭消费:总量']

plot.scatter(x13, y)

plot.xlabel("煤炭消费:总量")

plot.ylabel("碳排放量")

plot.show()

**因子分析：**

import pandas as pd

import numpy as np

import math as math

import numpy as np

from numpy import \*

from scipy.stats import bartlett

from factor\_analyzer import \*

import numpy.linalg as nlg

from sklearn.cluster import KMeans

from matplotlib import cm

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

def main():

df=pd.read\_excel("final(1).xlsx",index\_col='年份')

df2=df.copy()

print("\n原始数据:\n")

display(df2)

# 皮尔森相关系数

df2\_corr=df2.corr()

print("\n相关系数:\n")

display(df2\_corr)

#热力图

cmap = cm.Blues

fig=plt.figure(figsize=(12,12))

ax=fig.add\_subplot()

map = ax.imshow(df2\_corr, interpolation='nearest', cmap=cmap, vmin=0, vmax=1)

plt.title('correlation coefficient--headmap')

ax.set\_yticks(range(len(df2\_corr.columns)))

ax.set\_yticklabels(df2\_corr.columns)

ax.set\_xticks(range(len(df2\_corr)))

ax.set\_xticklabels(df2\_corr.columns)

plt.xticks(rotation=300)

plt.colorbar(map)

plt.show()

# KMO测度

def kmo(dataset\_corr):

corr\_inv = np.linalg.inv(dataset\_corr)

nrow\_inv\_corr, ncol\_inv\_corr = dataset\_corr.shape

A = np.ones((nrow\_inv\_corr, ncol\_inv\_corr))

for i in range(0, nrow\_inv\_corr, 1):

for j in range(i, ncol\_inv\_corr, 1):

A[i, j] = -(corr\_inv[i, j]) / (math.sqrt(corr\_inv[i, i] \* corr\_inv[j, j]))

A[j, i] = A[i, j]

dataset\_corr = np.asarray(dataset\_corr)

kmo\_num = np.sum(np.square(dataset\_corr)) - np.sum(np.square(np.diagonal(A)))

kmo\_denom = kmo\_num + np.sum(np.square(A)) - np.sum(np.square(np.diagonal(A)))

kmo\_value = kmo\_num / kmo\_denom

return kmo\_value

print("\nKMO测度:", kmo(df2\_corr))

# 巴特利特球形检验

df2\_corr1 = df2\_corr.values

print("\n巴特利特球形检验:", bartlett( df2\_corr1[1], df2\_corr1[2], df2\_corr1[3], df2\_corr1[4],

df2\_corr1[5], df2\_corr1[6], df2\_corr1[7], df2\_corr1[8], df2\_corr1[9],

df2\_corr1[10], df2\_corr1[11], df2\_corr1[12]))

# 求特征值和特征向量

eig\_value, eigvector = nlg.eig(df2\_corr)

eig = pd.DataFrame()

eig['names'] = df2\_corr.columns

eig['eig\_value'] = eig\_value

eig.sort\_values('eig\_value', ascending=False, inplace=True)

print("\n特征值：\n")

display(eig)

eig1=pd.DataFrame(eigvector)

eig1.columns = df2\_corr.columns

eig1.index = df2\_corr.columns

print("\n特征向量：\n")

display(eig1)

#使用前m个特征值的比重大于99%的标准，选出了三个公共因子

for m in range(1, 14):

if eig['eig\_value'][:m].sum()/ eig['eig\_value'].sum() >= 0.99:

print("\n公因子个数:", m)

break

# 因子载荷阵

A = np.mat(np.zeros((13, 3)))

i = 0

j = 0

while i < 3:

j = 0

while j < 13:

A[j:, i] = sqrt(eig\_value[i]) \* eigvector[j, i]

j = j + 1

i = i + 1

a = pd.DataFrame(A)

a.columns = ['factor1','factor2','factor3']

a.index = df2\_corr.columns

print("\n因子载荷阵：\n")

display(a)

fa = FactorAnalyzer(n\_factors=3)

fa.loadings\_ = a

print("\n特殊因子方差：\n")

display(pd.DataFrame(fa.get\_communalities()))

var = fa.get\_factor\_variance()

print("\n总方差（贡献率）：\n")

display(pd.DataFrame(var))

if \_\_name\_\_ == '\_\_main\_\_':

main()

**建模：**

from sklearn.linear\_model import LinearRegression

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import warnings

df=pd.read\_excel('final(1).xlsx',index\_col=0)

#未筛出主要因子前

from sklearn import model\_selection

from sklearn.metrics import mean\_squared\_error

X = df.iloc[:,:13]

y = df.iloc[:,13]

x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(X,y,test\_size=0.2,random\_state=1234)

linreg = LinearRegression()

linreg.fit(x\_train, y\_train)

#输出线性回归模型的截距和回归系数

print(pd.Series(index=['Intercept']+x\_train.columns.tolist(),

data=[linreg.intercept\_]+linreg.coef\_.tolist()))

#模型评估

linreg\_pred=linreg.predict(x\_test)

#均方误差

MSE=mean\_squared\_error(y\_test,linreg\_pred)

print('均方根误差{:.2f}'.format(np.sqrt(MSE)))

predict\_score=linreg.score(x\_test,y\_test)

print('评分{}'.format(predict\_score))

#筛出主要因子

X = df.iloc[:,:3]

y = df.iloc[:,13]

x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(X,y,test\_size=0.2,random\_state=1234)

linreg = LinearRegression()

linreg.fit(x\_train, y\_train)

#输出线性回归模型的截距和回归系数

print(pd.Series(index=['Intercept']+x\_train.columns.tolist(),

data=[linreg.intercept\_]+linreg.coef\_.tolist()))

#模型评估

linreg\_pred=linreg.predict(x\_test)

#均方误差

MSE=mean\_squared\_error(y\_test,linreg\_pred)

print('均方根误差{:.2f}'.format(np.sqrt(MSE)))

predict\_score=linreg.score(x\_test,y\_test)

print('评分{}'.format(predict\_score))

#根据发现问题单独将GDP和碳排放拟合

from sklearn.linear\_model import LassoCV

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

plt.rcParams['font.sans-serif'] = ['Simsun']

plt.rcParams['axes.unicode\_minus'] = False

warnings.filterwarnings('ignore')

x = np.array(df.loc[:,'国内生产总值']).reshape(-1, 1)

y = np.array(df.loc[:,'碳排放']).reshape(-1, 1)

model=Pipeline([('poly',PolynomialFeatures()),

('linear',LassoCV(alphas=np.logspace(-5,2,200),normalize=True,cv=10,max\_iter=45000))])

model.set\_params(poly\_\_degree=4)

model.fit(x,y)

lasso\_cv=model.get\_params('linear')['linear']

print(lasso\_cv.coef\_.ravel())

coef=[round(c,2)for c in lasso\_cv.coef\_.ravel()]

x\_hat=np.linspace(x.min(),x.max(),num=100).reshape(-1,1)

y\_hat=model.predict(x\_hat)

plt.scatter(x,y)

plt.plot(x\_hat,y\_hat,color='r')

plt.xlabel("GDP")

plt.ylabel("碳排放量")

plt.show()

#0-1分类（未筛出主要因子）

from keras.models import Sequential

from keras.layers.core import Dense, Activation

df['碳排放1']=df['碳排放'].diff()

df.loc[(df['碳排放1']<0), '碳排放2']=0

df.loc[(df['碳排放1']>=0), '碳排放2']=1

x=df.iloc[1:,0:13]

y=df.loc[1992:,'碳排放2']

model = Sequential()# 建立模型

model.add(Dense(input\_dim= 13, units = 10))

model.add(Activation('relu')) # 用relu函数作为激活函数，能够大幅提供准确度

model.add(Dense(input\_dim= 10, units = 1))

model.add(Activation('sigmoid')) # 由于是0-1输出，用sigmoid函数作为激活函数

model.compile(loss = 'binary\_crossentropy', optimizer = 'adam',metrics='accuracy')# 编译模型

model.fit(x, y, epochs = 1000, batch\_size= 10) # 训练模型，学习一千次

yp= model.predict(x).reshape(len(y)) # 分类预测

yp[yp>=0.5]=1

yp[yp<0.5]=0

score,acc=model.evaluate(x,y)

print(acc)

#0-1分类（筛出主要因子）

x=df.iloc[1:,0:3]

model = Sequential()# 建立模型

model.add(Dense(input\_dim= 3, units = 10))

model.add(Activation('relu')) # 用relu函数作为激活函数，能够大幅提供准确度

model.add(Dense(input\_dim= 10, units = 1))

model.add(Activation('sigmoid')) # 由于是0-1输出，用sigmoid函数作为激活函数

model.compile(loss = 'binary\_crossentropy', optimizer = 'adam',metrics='accuracy')# 编译模型

model.fit(x, y, epochs = 1000, batch\_size= 10) # 训练模型，学习一千次

yp= model.predict(x).reshape(len(y)) # 分类预测

yp[yp>=0.5]=1

yp[yp<0.5]=0

score,acc=model.evaluate(x,y)

print(acc)

#bp神经网络手撕代码

#学习函数

def logsig(x):

return 1/(1+np.exp(-x))

#数据

def loaddataset(filename):

df = pd.read\_excel(filename,index\_col=0)

#存放数据

dataset = []

#存放标签

labelset = []

for i in df.index:

a = list(df.loc[i,:])

#每个数据行的最后一个是标签

dataset.append([float(j) for j in a[:len(a)-1]])

labelset.append(int(float(a[-1])))

return dataset, labelset

def loaddataset1(filename):

df = pd.read\_excel(filename,index\_col=0)

#存放数据

dataset = []

#存放标签

labelset = []

for i in df.index:

a = list(df.loc[i,:])

#每个数据行的最后一个是标签

dataset.append([float(j) for j in a[0:3]])

labelset.append(int(float(a[-1])))

return dataset, labelset

#BP神经网络

def bp(dataset,labelset):

#数据转为矩阵

samplein=np.mat(dataset)

sampleinminmax=np.array([samplein.min(axis=1).T.tolist()[0],samplein.max(axis=1).T.tolist()[0]]).transpose() #3\*2

sampleout=np.mat(labelset)

sampleoutminmax=np.array([sampleout.min(axis=1).T.tolist()[0],sampleout.max(axis=1).T.tolist()[0]]).transpose() #2\*2

#最大最小法归一

sampleinnorm=(2\*(np.array(samplein.T)-sampleinminmax.transpose()[0])/(sampleinminmax.transpose()[1]-sampleinminmax.transpose()[0])-1).transpose()

sampleoutnorm=(2\*(np.array(sampleout.T)-sampleoutminmax.transpose()[0])/(sampleoutminmax.transpose()[1]-sampleoutminmax.transpose()[0])-1).transpose()

#给输出样本添加噪音

noise=0.03\*np.random.rand(sampleoutnorm.shape[0])

sampleoutnorm+=noise

#定义模型参数

maxepochs=30000

learnrate=0.035 #学习效率

errorfinal=0.65\*10\*\*(-3) #误差阈值

samnum=29 #样本个数

indim=3 #输入变量个数

outdim=1#输出变量个数

hiddenunitnum=8 #隐藏层神经元个数

w1=0.5\*np.random.rand(hiddenunitnum,indim)-0.1

b1=0.5\*np.random.rand(hiddenunitnum,1)-0.1

w2=0.5\*np.random.rand(outdim,hiddenunitnum)-0.1

b2=0.5\*np.random.rand(outdim,1)-0.1

#训练模型

errhistory=[]

for i in range(maxepochs):

hiddenout=logsig((np.dot(w1,sampleinnorm.transpose()).transpose()+b1.transpose())).transpose() #隐含层结果

networkout=(np.dot(w2,hiddenout).transpose()+b2.transpose()).transpose() #输出层结果

err=sampleoutnorm-networkout

sse=sum(sum(err\*\*2))

errhistory.append(sse)

if sse<errorfinal:

break

delta2=err

delta1=np.dot(w2.transpose(),delta2)\*hiddenout\*(1-hiddenout)

dw2=np.dot(delta2,hiddenout.transpose())

db2=np.dot(delta2,np.ones((samnum,1)))

dw1=np.dot(delta1,sampleinnorm)

db1=np.dot(delta1,np.ones((samnum,1)))

w2+=learnrate\*dw2

b2+=learnrate\*db2

w1+=learnrate\*dw1

b1+=learnrate\*db1

#误差曲线

errhistory10=np.log10(errhistory)

minerr=min(errhistory10)

plt.plot(errhistory10)

plt.plot(range(0,i+1000,1000),[minerr]\*len(range(0,i+1000,1000)))

ax=plt.gca() #获得当前对象ax

#自定义刻度线标签

ax.set\_yticks([-2,-1,0,1,2,minerr])

ax.set\_yticklabels([u'$10^{-2}$',u'$10^{-1}$',u'$1$',u'$10^{1}$',u'$10^{2}$',str(('%.4f'%np.power(10,minerr)))])

ax.set\_xlabel('iteration')

ax.set\_ylabel('error')

ax.set\_title('Error History')

#plt.savefig("errorhistory.png",dpi=700) #保存图片

plt.show()

plt.close()

df=pd.read\_excel('final(1).xlsx',index\_col=0)

#存放数据

dataset = []

#存放标签

labelset = []

for i in df.index:

a = list(df.loc[i,:])

#每个数据行的最后一个是标签

dataset.append([float(j) for j in a[:3]])

labelset.append(int(float(a[-1])))

bp(dataset,labelset)

**北半球气温预测：**

import pyecharts.options as opts

from pyecharts.charts import Line

import numpy as np

import pandas as pd

from datetime import datetime

df = pd.read\_excel('北半球气温.xls', index\_col='Day')

x =pd.to\_datetime(df.index,format='%Y-%m-%d')

print(x)

ts = df['temperature\_anomaly']

train = ts[0:1440]

trainx=pd.to\_datetime(train.index,format='%Y-%m-%d')

test = ts[1440:]

testx=pd.to\_datetime(test.index,format='%Y-%m-%d')

c = (

Line()

.add\_xaxis(x)

.add\_yaxis("Total",ts,symbol\_size= 0)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title='北半球气温'),

tooltip\_opts=opts.TooltipOpts(

is\_show = True,

trigger="axis",

axis\_pointer\_type="cross"

),

yaxis\_opts=opts.AxisOpts(

type\_="value",

axistick\_opts=opts.AxisTickOpts(is\_show=True),

splitline\_opts=opts.SplitLineOpts(is\_show=True),

),

xaxis\_opts=opts.AxisOpts(

type\_="category",

boundary\_gap=False,

),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

.render('北半球气温.html')

)

import pyecharts.options as opts

from pyecharts.charts import Line

from statsmodels.tsa.api import ExponentialSmoothing

fit1 = ExponentialSmoothing(np.asarray(train) ,seasonal\_periods=12 ,trend='add', seasonal='add').fit()

test1= fit1.forecast(len(test))

test1=pd.Series(test1,index=test.index)

c = (

Line()

.add\_xaxis(test.index)

.add\_yaxis("Test",test,symbol\_size= 0)

.add\_yaxis("Holt\_Winter",test1,symbol\_size= 0)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title='北半球气温预测（基于历史数据）'),

tooltip\_opts=opts.TooltipOpts(

is\_show = True,

trigger="axis",

axis\_pointer\_type="cross"

),

yaxis\_opts=opts.AxisOpts(

type\_="value",

axistick\_opts=opts.AxisTickOpts(is\_show=True),

splitline\_opts=opts.SplitLineOpts(is\_show=True),

),

xaxis\_opts=opts.AxisOpts(

type\_="category",

boundary\_gap=False,

),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

.render('北半球气温预测（基于历史数据）.html')

)

test\_future=pd.date\_range(start=datetime.now().strftime('%Y-%m-%d'),periods=240,freq='m')

fit2 = ExponentialSmoothing(np.asarray(test) ,seasonal\_periods=12 ,trend='add', seasonal='add').fit()

test2= fit2.forecast(len(test\_future))

test2=pd.Series(test2,index=test\_future)

c = (

Line()

.add\_xaxis(test\_future)

.add\_yaxis("Holt\_Winter",test2,symbol\_size= 0)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title='北半球气温预测2'),

tooltip\_opts=opts.TooltipOpts(

is\_show = True,

trigger="axis",

axis\_pointer\_type="cross"

),

yaxis\_opts=opts.AxisOpts(

type\_="value",

axistick\_opts=opts.AxisTickOpts(is\_show=True),

splitline\_opts=opts.SplitLineOpts(is\_show=True),

),

xaxis\_opts=opts.AxisOpts(

type\_="category",

boundary\_gap=False,

),

)

.set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))

.render('北半球气温预测2.html')

)